**Exercise 2: E-commerce Platform Search Function**

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Solution**

**Product.java**

public class Product {

    private int id;

    private String name;

    private double price;

    public Product(int id, String name, double price) {

        this.id = id;

        this.name = name.toLowerCase(); // Normalize for better search

        this.price = price;

    }

    public int getId() {

        return id;

    }

    public String getName() {

        return name;

    }

    public double getPrice() {

        return price;

    }

    @Override

    public String toString() {

        return "Product [ID=" + id + ", Name=" + name + ", Price=" + price + "]";

    }

}

**SearchFunctions.java**

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

public class SearchFunctions {

    // Sample product list

    private List<Product> productList;

    public SearchFunctions() {

        productList = new ArrayList<>();

    }

    public void addProduct(Product product) {

        productList.add(product);

    }

    // Fast search using Java Streams (optimized for name contains keyword)

    public List<Product> searchByName(String keyword) {

        String searchKey = keyword.toLowerCase();

        return productList.stream()

                .filter(p -> p.getName().contains(searchKey))

                .collect(Collectors.toList());

    }

    // Search by exact product ID

    public Product searchById(int id) {

        for (Product p : productList) {

            if (p.getId() == id) {

                return p;

            }

        }

        return null;

    }

}

**SearchTest.java**

import java.util.List;

public class SearchTest {

    public static void main(String[] args) {

        SearchFunctions search = new SearchFunctions();

        // Adding sample products

        search.addProduct(new Product(1, "Apple iPhone", 69999));

        search.addProduct(new Product(2, "Samsung Galaxy", 59999));

        search.addProduct(new Product(3, "Redmi Note", 19999));

        search.addProduct(new Product(4, "Apple Watch", 29999));

        // Search by name

        System.out.println("Search Results for 'apple':");

        List<Product> results = search.searchByName("apple");

        for (Product p : results) {

            System.out.println(p);

        }

        // Search by ID

        System.out.println("\nSearch Product with ID = 3:");

        Product found = search.searchById(3);

        if (found != null) {

            System.out.println(found);

        } else {

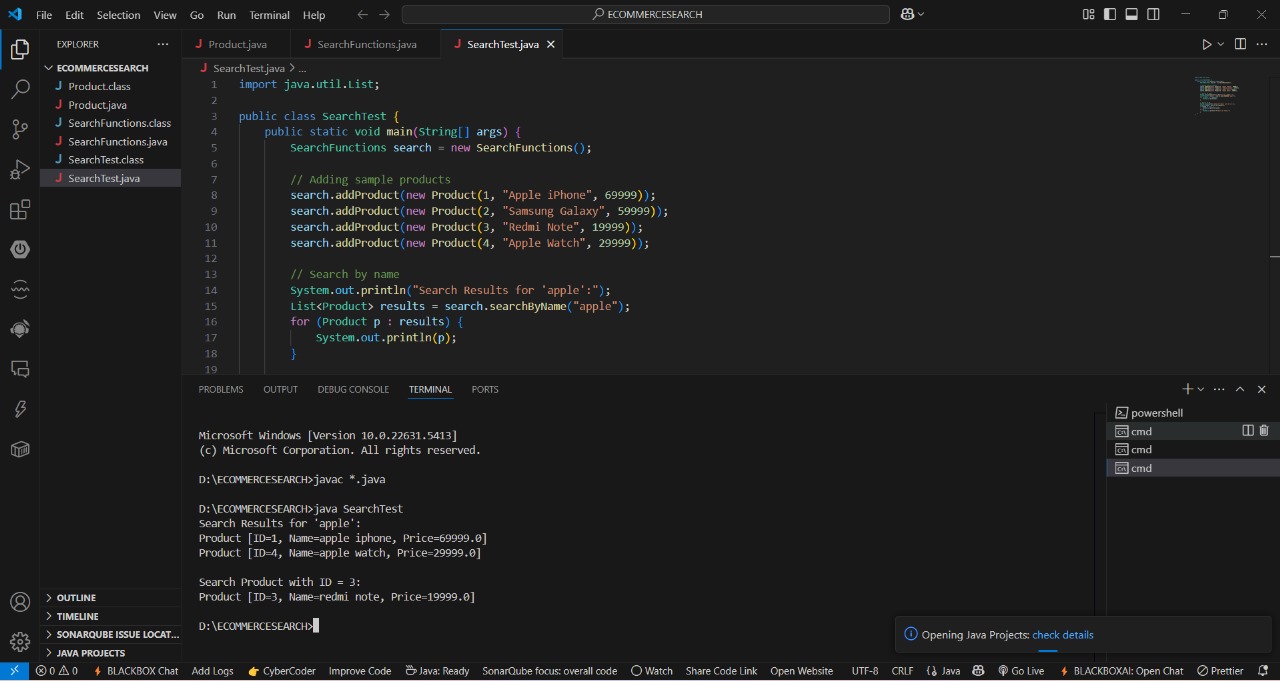
            System.out.println("Product not found.");

        }

    }

}

**Output:**



**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past dat

**Solution:**

**FinancialForecast.java**

public class FinancialForecast {

    // Function to calculate the mean

    public static double mean(double[] data) {

        double sum = 0;

        for (double d : data) {

            sum += d;

        }

        return sum / data.length;

    }

    // Function to calculate linear regression coefficients

    public static double[] linearRegression(double[] x, double[] y) {

        int n = x.length;

        double meanX = mean(x);

        double meanY = mean(y);

        double numerator = 0, denominator = 0;

        for (int i = 0; i < n; i++) {

            numerator += (x[i] - meanX) \* (y[i] - meanY);

            denominator += (x[i] - meanX) \* (x[i] - meanX);

        }

        double m = numerator / denominator;

        double c = meanY - m \* meanX;

        return new double[]{m, c};

    }

    // Predict the value for the next time period

    public static double predictNextValue(double[] x, double[] y, double nextX) {

        double[] coefficients = linearRegression(x, y);

        double m = coefficients[0];

        double c = coefficients[1];

        return m \* nextX + c;

    }

    public static void main(String[] args) {

        // Example past data: months (x) and revenue (y)

        double[] months = {1, 2, 3, 4, 5}; // e.g., Jan to May

        double[] revenue = {10000, 12000, 13000, 15000, 16000};

        double nextMonth = 6; // Predicting for June

        double forecast = predictNextValue(months, revenue, nextMonth);

        System.out.printf("Forecasted revenue for month %.0f: ₹%.2f%n", nextMonth, forecast);

    }

}

**Output:**

